Lukasiewicz mu-Calculus

Łukasiewicz μ-calculus

Matteo Mio
CWI, Amsterdam (NL)
miomatteo@gmail.com

Alex Simpson
LFCS, School of Informatics
University of Edinburgh
Alex.Simpson@ed.ac.uk

The paper explores properties of *Łukasiewicz μ-calculus*, a version of the quantitative/probabilistic modal μ-calculus containing both weak and strong conjunctions and disjunctions from Łukasiewicz (fuzzy) logic. We show that this logic encodes the well-known probabilistic temporal logic **PCTL**. And we give a model-checking algorithm for computing the rational denotational value of a formula at any state in a finite rational probabilistic nondeterministic transition system.

1 Introduction

Among logics for expressing properties of nondeterministic (including concurrent) processes, represented as transition systems, Kozen's modal μ-calculus [15] plays a fundamental rôle. It subsumes other temporal logics of processes, such as **LTL**, **CTL** and **CTL***. It does not distinguish bisimilar processes, but separates (finite) non-bisimilar ones. More generally, by a remarkable result of Janin and Walukiewicz [14], it is exactly as expressive as the bisimulation-invariant fragment of monadic second-order logic. Furthermore, there is an intimate connection with parity games, which offers an intuitive reading of fixed-points, and underpins the existing technology for model-checking μ-calculus properties.

For many purposes, it is useful to add probability to the computational model, leading to probabilistic nondeterministic transition systems, cf. [23]. Among the different approaches that have been followed to developing analogues of the modal μ-calculus in this setting, the most significant is that introduced independently by Huth and Kwiatkowska [12] and by Morgan and McIver [22], under which a *quantitative* interpretation is given, with formulas denoting values in $[0,1]$. This quantitative setting permits several variations. In particular, three different quantitative extensions of conjunction from booleans to $[0,1]$ (with 0 as false and 1 as true) arise naturally [12]: minimum, $\min(x,y)$; multiplication, xy; and the strong conjunction (a.k.a. Łukasiewicz t-norm) from Łukasiewicz fuzzy logic, $\max(x+y-1,\,0)$. In each case, there is a dual operator giving a corresponding extension of disjunction: maximum, $\max(x,y)$; comultiplication, $x+y-xy$; and Łukasiewicz strong disjunction, $\min(x+y,\,1)$. The choice of min and max for conjunction and disjunction is particularly natural, since the corresponding μ-calculus, called qLμ in [18], has an interpretation in terms of 2-player *stochastic* parity games, which extends the usual parity-game interpretation of the ordinary modal μ-calculus. This allows the real number denoted by a formula to be understood as the *value* of the associated game [18, 20].

The present paper contributes to a programme of ongoing research, one of whose overall aims is to investigate the extent to which quantitative μ-calculi play as fundamental a rôle in the probabilistic setting as that of Kozen's μ-calculus in the nondeterministic setting. The logic qLμ, with min/max as conjunction/disjunction, is insufficiently expressive. For example, it cannot encode the standard probabilistic temporal logic **PCTL** of [2]. Nevertheless, richer calculi can be obtained by augmenting qLμ with the other alternatives for conjunction/disjunction, to be used in combination with max and min. Such extensions were investigated by the first author in [21, 19], where the game-theoretic interpretation was generalized to accommodate the new operations.

D. Baelde and A. Carayol (Eds.): Fixed Points
in Computer Science 2013 (FICS 2013)
EPTCS 126, 2013, pp. 87–104, doi:10.4204/EPTCS.126.7

In this paper, we focus on a calculus containing two different interpretations of conjunction and disjunction: min and max (written as $\sqcap$ and $\sqcup$) and the Łukasiewicz operations (written as $\odot$ and $\oplus$). In addition, as is natural in the quantitative setting, we include a basic operation for multiplying the value of a formula by a rational constant in $[0,1]$. Since these operations are all familiar from Łukasiewicz fuzzy logic (see, e.g., [11]), we call the resulting logic *Łukasiewicz μ-calculus* (Łμ).

As our first contribution, we show that the standard probabilistic temporal logic **PCTL** [2] can be encoded in Łμ. A similar translation was originally given in the first author's PhD thesis [19], where **PCTL** was translated into a quantitative μ-calculus containing all three pairs of quantitative conjunction/disjunction operations in combination. Here, we streamline the treatment by implementing the observation that the (co)multiplication operations are not required once the Łukasiewicz operations are in place. In fact, all that is needed is the encodability of certain *threshold modalities*, see Remark 3.6 below.

An advantage of the Łukasiewicz μ-calculus considered in the present paper is that it enjoys the property that the value of a formula in a finite rational model is rational, a property which does not hold when the (co)multiplication operations are included in the logic. As our second contribution, we exploit this property by giving a (quantitative) model-checking algorithm that computes the value of a Łμ formula at a state in a finite rational probabilistic nondeterministic transition system. The algorithm adapts the approximation-based approach to nested fixed-point calculation to our quantitative calculus.

One could combine our two contributions and obtain a new model-checking algorithm for **PCTL**. But this is not advisable since the complexity bounds we obtain for model-checking Łμ are abysmal. The positive messages of this paper are rather that **PCTL** fits into the conceptually appealing framework of quantitative μ-calculi, and that this framework is itself algorithmically approachable.

2 Technical background

Definition 2.1. Given a set S we denote with $\mathscr{D}(S)$ the set of *(discrete) probability distributions* on S defined as $\mathscr{D}(S) = \{d : S \to [0,1] \mid \sum_{s\in S} d(s) = 1\}$. We say that $d \in \mathscr{D}(S)$ is *rational* if $d(s)$ is a rational number, for all $s \in S$.

Definition 2.2. A *probabilistic nondeterministic transition system* (PNTS) is a pair $(S, \to)$ where S is a set of states and $\to \subseteq S \times \mathscr{D}(S)$ is the *accessibility* relation. We write $s \not\to$ if $\{d \mid s \to d\} = \emptyset$. A PNTS $(S, \to)$ is *finite rational* if S is finite and $\bigcup_{s\in S}\{d \mid s \to d\}$ is a finite set of rational probability distributions.

We now introduce the novel logic Łμ which extends the probabilistic (or quantitative) modal μ-calculus (qLμ) of [12, 22, 18, 5].

Definition 2.3. The logic Łμ is generated by the following grammar:

$$\phi ::= X \mid P \mid \overline{P} \mid q\phi \mid \phi \sqcup \phi \mid \phi \sqcap \phi \mid \phi \oplus \phi \mid \phi \odot \phi \mid \Diamond\phi \mid \Box\phi \mid \mu X.\phi \mid \nu X.\phi \ ,$$

where q ranges over rationals in $[0,1]$, X over a countable set `Var` of variables and P over a set `Prop` of propositional letters which come paired with associated complements $\overline{P}$. As a convention we denote with $\underline{1}$ the formula $\nu X.X$ and with $\underline{q}$ the formula $q\underline{1}$.

Thus, Łμ extends the syntax of the probabilistic modal μ-calculus by the new pair of connectives $(\odot, \oplus)$, which we refer to as *Łukasiewicz conjunction* and *disjunction*, respectively, and a form of *scalar multiplication* $(q\phi)$ by rationals numbers in $[0,1]$. For mild convenience in the encoding of **PCTL** below, we consider a version with unlabelled modalities and propositional letters. However, the approach of this paper easily adapts to a labeled version of Łμ.

Formulas are interpreted over PNTS's as we now describe.

Definition 2.4. Given a PNTS $(S,\to)$, an *interpretation* for the variables and propositional letters is a function $\rho : (\mathtt{Var}\uplus\mathtt{Prop}) \to (S\to[0,1])$ such that $\rho(\overline{P})(x) = 1-\rho(P)(x)$. Given a function $f : S\to[0,1]$ and $X\in\mathtt{Var}$ we define the interpretation $\rho[f/X]$ as $\rho[f/X](X) = f$ and $\rho[f/X](Y) = \rho(Y)$, for $X\neq Y$.

Definition 2.5. The semantics of a Łμ formula ϕ interpreted over $(S,\to)$ with interpretation ρ is a function $[\![\phi]\!]_\rho : S\to[0,1]$ defined inductively on the structure of ϕ as follows:

$$[\![X]\!]_\rho = \rho(X) \qquad [\![q\,\phi]\!]_\rho(x) = q\cdot[\![\phi]\!]_\rho(x)$$
$$[\![P]\!]_\rho = \rho(P) \qquad [\![\overline{P}]\!]_\rho = 1-\rho(P)$$
$$[\![\phi\sqcup\psi]\!]_\rho(x) = \max\{[\![\phi]\!]_\rho(x),[\![\psi]\!]_\rho(x)\} \qquad [\![\phi\sqcap\psi]\!]_\rho(x) = \min\{[\![\phi]\!]_\rho(x),[\![\psi]\!]_\rho(x)\}$$
$$[\![\phi\oplus\psi]\!]_\rho(x) = \min\{1,[\![\phi]\!]_\rho(x)+[\![\psi]\!]_\rho(x)\} \qquad [\![\phi\odot\psi]\!]_\rho(x) = \max\{0,[\![\phi]\!]_\rho(x)+[\![\psi]\!]_\rho(x)-1\}$$
$$[\![\Diamond\phi]\!]_\rho(x) = \bigsqcup_{x\to d}\Big(\sum_{y\in X} d(y)[\![\phi]\!]_\rho(y)\Big) \qquad [\![\Box\phi]\!]_\rho(x) = \bigsqcap_{x\to d}\Big(\sum_{y\in X} d(y)[\![\phi]\!]_\rho(y)\Big)$$
$$[\![\mu X.\phi]\!] = \mathrm{lfp}\big(f\mapsto[\![\phi]\!]_{\rho[f/X]}\big) \qquad [\![\mu X.\phi]\!] = \mathrm{gfp}\big(f\mapsto[\![\phi]\!]_{\rho[f/X]}\big)$$

It is straightforward to verify that the interpretation of every operator is monotone, thus the existence of least and greatest points in the last two clauses is guaranteed by the the Knaster-Tarski theorem.

As customary in fixed-point logics, we presented the logic Łμ in positive normal form. A negation operation $\mathtt{dual}(\phi)$ can be defined on *closed* formulas by replacing every connective with its dual and $(q\,\phi)$ with $((1-q)\,\phi)$. It is simple to verify that $[\![\mathtt{dual}(\phi)]\!]_\rho(x) = 1-[\![\phi]\!]_\rho(x)$.

Next, we introduce the syntax and the semantics of the logic **PCTL** of [2]. We refer to [1] for an extensive presentation of this logic.

The notions of *paths*, *schedulers* and *Markov runs* in a PNTS are at the basis of the logic **PCTL**.

Definition 2.6. For a given PNTS $\mathscr{L} = (S,\to)$ the binary relation $\rightsquigarrow_{\mathscr{L}}\subseteq S\times S$ is defined as follows: $\rightsquigarrow_{\mathscr{L}} = \{(s,t)\mid\exists d.(s\to d\ \wedge\ d(t)>0)\}$. Note that $s\not\to$ if and only if $s\not\rightsquigarrow$. We refer to $(S,\rightsquigarrow)$ as the *graph underlying* $\mathscr{L}$.

Definition 2.7. A *path* in a PNTS $\mathscr{L} = (S,\to)$ is an ordinary path in the graph $(S,\rightsquigarrow)$, i.e., a finite or infinite sequence $\{s_i\}_{i\in I}$ of states such that $s_i\rightsquigarrow s_{i+1}$, for all $i+1\in I$. We say that a path is *maximal* if either it is infinite or it is finite and its last entry is a state s_n without successors, i.e., such that $s_n\not\rightsquigarrow$. We denote with $\mathrm{P}(\mathscr{L})$ the set of all maximal paths in $\mathscr{L}$. The set $\mathrm{P}(\mathscr{L})$ is endowed with the topology generated by the basic open sets $U_{\vec{s}} = \{\vec{r}\mid\vec{s}\sqsubseteq\vec{r}\}$ where $\vec{s}$ is a finite sequence of states and $\sqsubseteq$ denotes the prefix relation on sequences. The space $\mathrm{P}(\mathscr{L})$ is always 0-dimensional, i.e., the basic sets $U_{\vec{s}}$ are both open and closed and thus form a Boolean algebra. We denote with $\mathrm{P}(s)$ the open set $U_{\{s\}}$ of all maximal paths having s as first state.

Definition 2.8. A *scheduler* in a PNTS $(S,\to)$ is a partial function σ from non-empty finite sequences $s_0\ldots s_n$ of states to probability distributions $d\in\mathscr{D}(S)$ such that $\sigma(s_0\ldots s_n)$ is not defined if and only if $s_n\not\to$ and, if σ is defined at $s_0\ldots s_n$ with $\sigma(s_0\ldots s_n) = d$, then $s_n\to d$ holds. A pair (s,σ) is called a *Markov run* in $\mathscr{L}$ and denoted by M^s_σ. It is clear that each Markov run M^s_σ can be identified with a (generally) infinite Markov chain (having a tree structure) whose vertices are finite sequences of states and having $\{s\}$ as root.

Markov runs are useful as they naturally induce probability measures on the space $\mathrm{P}(\mathscr{L})$.

Definition 2.9. Let $\mathscr{L} = (S,\to)$ be a PNTS and M^s_σ a Markov run. We define the measure m^s_σ on $\mathrm{P}(\mathscr{L})$ as the unique (by Carathéodory extension theorem) measure specified by the following assignment of basic open sets:

$$m^s_\sigma\big(\mathrm{U}_{s_0\ldots s_n}\big) = \prod_{i=0}^{n-1} d_i(s_{i+1})$$

where $d_i = \sigma(s_0 \dots s_i)$ and $\prod \emptyset = 1$. It is simple to verify that m^s_σ is a probability measure, i.e., $m^s_\sigma(\mathrm{P}(\mathscr{L})) = 1$. We refer to m^s_σ as the probability measure on $\mathrm{P}(\mathscr{L})$ induced by the Markov run M^s_σ.

We are now ready to specify the syntax and semantics of **PCTL**.

Definition 2.10. Let the letter P range over a countable set of propositional symbols `Prop`. The class of **PCTL** *state-formulas* ϕ is generated by the following two-sorted grammar:

$$\phi ::= \text{true} \mid P \mid \neg\phi \mid \phi \vee \phi \mid \exists\psi \mid \forall\psi \mid \mathbb{P}^{\exists}_{\bowtie q}\psi \mid \mathbb{P}^{\forall}_{\bowtie q}\psi$$

with $q \in \mathbb{Q} \cap [0,1]$ and $\bowtie \in \{>, \geq\}$, where *path-formulas* ψ are generated by the simple grammar: $\psi ::= \circ\phi \mid \phi_1 \mathscr{U} \phi_2$. Adopting standard terminology, we refer to the connectives $\circ$ and $\mathscr{U}$ as the *next* and *until* operators, respectively.

Definition 2.11. Given a PNTS $(S, \to)$, a ***PCTL*-*interpretation*** for the propositional letters is a function $\rho : \mathtt{Prop} \to 2^S$, where 2^S denotes the powerset of S.

Definition 2.12. Given a PNTS $(S, \to)$ and a **PCTL**-interpretation ρ for the propositional letters, the semantics $(\!|\phi|\!)_\rho$ of a **PCTL** state-formula ϕ is a subset of S (i.e., $(\!|\phi|\!)_\rho : S \to \{0,1\}$) defined by induction on the structure of ϕ as follows:

- $(\!|\text{true}|\!)_\rho = S$, $(\!|P|\!)_\rho = \rho(P)$, $(\!|\phi_1 \vee \phi_2|\!)_\rho = (\!|\phi_1|\!)_\rho \cup (\!|\phi_2|\!)_\rho$, $(\!|\neg\phi|\!)_\rho = S \setminus (\!|\phi|\!)_\rho$,
- $(\!|\exists\psi|\!)_\rho(s) = 1$ if and only there exists $\vec{s} \in \mathrm{P}(s)$ such that that $\vec{s} \in [\![\psi]\!]$
- $(\!|\forall\psi|\!)_\rho(s) = 1$ if and only forall $\vec{s} \in \mathrm{P}(s)$ it holds that $\vec{s} \in (\!|\psi|\!)_\rho(\vec{s})$
- $(\!|\mathbb{P}^{\exists}_{\bowtie q}\psi|\!)_\rho(s) = 1$ if and only $\left(\bigsqcup_\sigma m^s_\sigma((\!|\psi|\!)_\rho)\right) \bowtie q$
- $(\!|\mathbb{P}^{\forall}_{\bowtie q}\psi|\!)_\rho(s) = 1$ if and only $\left(\bigsqcap_\sigma m^s_\sigma((\!|\psi|\!)_\rho)\right) \bowtie q$

where σ ranges over schedulers and the semantics $(\!|\psi|\!)_\rho$ of path formulas, defined as a subset of $\mathrm{P}(\mathscr{L})$ (i.e., as a map $(\!|\psi|\!)_\rho : \mathrm{P}(\mathscr{L}) \to \{0,1\}$) is defined as:

- $(\!|\circ\phi|\!)_\rho(\vec{s}) = 1$ if and only if $|\vec{s}| \geq 2$ (i.e., $\vec{s} = s_0.s_1\dots$) and $s_1 \in (\!|\phi|\!)_\rho$,
- $(\!|\phi_1 \mathscr{U} \phi_2|\!)_\rho(\vec{s}) = 1$ if and only if $\exists n.\big((s_n \in (\!|\phi_2|\!)_\rho) \wedge \forall m < n.(s_m \in (\!|\phi_1|\!)_\rho)\big)$,

It is simple to verify that, for all path-formulas ψ, the set $(\!|\psi|\!)_\rho$ is Borel measurable [1]. Therefore the definition is well specified. Note how the logic **PCTL** can express probabilistic properties, by means of the connectives $\mathbb{P}^{\forall}_{\bowtie q}$ and $\mathbb{P}^{\exists}_{\bowtie q}$, as well as (qualitative) properties of the graph underlying the PNTS by means of the quantifiers $\forall$ and $\exists$.

3 Encoding of PCTL

We prove in this section how **PCTL** can be seen as a simple fragment of Łμ by means of an explicit encoding. We first introduce a few useful macro formulas in the logic Łμ which, crucially, are not expressible in the probabilistic μ-calculus (qLμ).

Definition 3.1. Let ϕ be a (possibly open) Łμ formula. We define:

• $\mathbb{P}_{>0}\phi = \mu X.(X \oplus \phi)$ • $\mathbb{P}_{=1}\phi = \nu X.(X \odot \phi)$ • $\mathbb{P}_{>q}\phi = \mathbb{P}_{>0}(\phi \odot \underline{1-q})$ • $\mathbb{P}_{\geq q}\phi = \mathbb{P}_{=1}(\phi \oplus \underline{1-q})$

for $q \in \mathbb{Q} \cap (0,1)$. We write $\mathbb{P}_{\bowtie q}\phi$, for $q \in \mathbb{Q} \cap [0,1]$, to denote one of the four cases.

The following proposition describes the denotational semantics of these macro formulas.

Proposition 3.2. *Let $(S, \to)$ be a PNTS, ϕ a Łμ formula and ρ an interpretation of the variables. Then it holds that:*

$$[\![\mathbb{P}_{\bowtie q}\phi]\!]_\rho(s) = \begin{cases} 1 & \text{if } [\![\phi]\!]_\rho(s) \bowtie q \\ 0 & \text{otherwise} \end{cases}$$

Proof. For the case $\mathbb{P}_{>0}\phi$, observe that the map $x \mapsto q \oplus x$, for a fixed $q \in [0,1]$, has 1 as unique fixed point when $q > 0$, and 0 as the least fixed point when $q = 0$. The result then follows trivially. Similarly for $\mathbb{P}_{=1}\phi$. The other cases are trivial. □

The following lemma is also useful.

Lemma 3.3. *Let $(S, \to)$ be a PNTS, ϕ a Łμ formula and ρ an interpretation of the variables. Then:*

- $[\![\mathbb{P}_{>0}(\Diamond X)]\!]_\rho(s) = 1$ *iff* $\exists t.\big(s \rightsquigarrow t \wedge \rho(X)(t) > 0\big)$
- $[\![\mathbb{P}_{=1}(\Box X)]\!]_\rho(s) = 1$ *iff* $\forall t.\big(s \rightsquigarrow t \to \rho(X)(t) = 1\big)$

Proof. Note that $[\![\Diamond X]\!]_\rho(s) > 0$ iff there exists $s \to d$ such that $\sum_{t \in S} d(t)\rho(X)(t) > 0$ holds. This is the case iff $d(t) > 0$ (i.e., $s \rightsquigarrow t$) and $\rho(X)(t) > 0$, for some $t \in S$. The result then follows by Proposition 3.2. The case for $\mathbb{P}_{=1}(\Box X)$ is similar. □

Remark 3.4. When considering $\{0,1\}$-valued interpretations for X, the macro formula $\mathbb{P}_{>0}\Diamond$ expresses the meaning of the diamond modality in classical modal logic with respect to the graph $(S, \rightsquigarrow)$ underlying the PNTS. Similarly, $\mathbb{P}_{=1}\Box$ corresponds to the the classical box modality.

We are now ready to define the encoding of **PCTL** into Łμ.

Definition 3.5. We define the encoding $\mathbf{E}$ from **PCTL** formulas to closed Łμ formulas (where $\boxdot\phi$ stands for the Łμ formula $\Box\phi \sqcap \Diamond\underline{1}$), by induction on the structure of the **PCTL** formulas ϕ as follows:

1. $\mathbf{E}(P) = P$,
2. $\mathbf{E}(\text{true}) = \underline{1}$,
3. $\mathbf{E}(\phi_1 \vee \phi_2) = \mathbf{E}(\phi_1) \sqcup \mathbf{E}(\phi_2)$,
4. $\mathbf{E}(\neg\phi) = \text{dual}(\mathbf{E}(\phi))$,
5. $\mathbf{E}(\exists(\circ\phi)) = \mathbb{P}_{>0}\big(\Diamond\mathbf{E}(\phi)\big)$,
6. $\mathbf{E}(\forall(\circ\phi)) = \mathbb{P}_{=1}\big(\boxdot\mathbf{E}(\phi)\big)$,
7. $\mathbf{E}(\exists(\phi_1 \mathscr{U} \phi_2)) = \mu X.\Big(\mathbf{E}(\phi_2) \sqcup \big(\mathbf{E}(\phi_1) \sqcap \mathbb{P}_{>0}(\Diamond X)\big)\Big)$,
8. $\mathbf{E}(\forall(\phi_1 \mathscr{U} \phi_2)) = \mu X.\Big(\mathbf{E}(\phi_2) \sqcup \big(\mathbf{E}(\phi_1) \sqcap \mathbb{P}_{=1}(\boxdot X)\big)\Big)$,
9. $\mathbf{E}(\mathbb{P}^{\exists}_{\bowtie q}(\circ\phi)) = \mathbb{P}_{\bowtie q}\big(\Diamond\mathbf{E}(\phi)\big)$,
10. $\mathbf{E}(\mathbb{P}^{\forall}_{\bowtie q}(\circ\phi)) = \mathbb{P}_{\bowtie q}\big(\boxdot\mathbf{E}(\phi)\big)$,
11. $\mathbf{E}(\mathbb{P}^{\exists}_{\bowtie q}(\phi_1 \mathscr{U} \phi_2)) = \mathbb{P}_{\bowtie q}\Big(\mu X.\Big(\mathbf{E}(\phi_2) \sqcup \big(\mathbf{E}(\phi_1) \sqcap \Diamond X\big)\Big)\Big)$,
12. $\mathbf{E}(\mathbb{P}^{\forall}_{\bowtie q}(\phi_1 \mathscr{U} \phi_2)) = \mathbb{P}_{\bowtie q}\Big(\mu X.\Big(\mathbf{E}(\phi_2) \sqcup \big(\mathbf{E}(\phi_1) \sqcap \boxdot X\big)\Big)\Big)$,

Note that Case 4 is well defined since $\mathbf{E}(\phi)$ is closed by construction.

Remark 3.6. The only occurrences of Łukasiewicz operators $\{\oplus,\odot\}$ and scalar multiplication $(q\,\phi)$ in encoded **PCTL** formulas appear in the formation of the macro formulas $\mathbb{P}_{\bowtie q}(\_)$ which we refer to as *threshold modalities*. Thus, **PCTL** can be also seen as a fragment of qLμ extended with threshold modalities as primitive operations. With the aid of these modalities the encoding is, manifestly, a straightforward adaption of the standard encoding of CTL into the modal μ-calculus (see, e.g., [24]).

We are now ready to prove the correctness theorem which holds for arbitrary models.

Theorem 3.7. *For every PNTS $(S,\to)$, **PCTL**-interpretation $\rho:\mathrm{Prop}\to(S\to\{0,1\})$ of the propositional letters and **PCTL** formula ϕ, the equality $(\!|\phi|\!)_\rho(s)=[\![\mathbf{E}(\phi)]\!]_\rho(s)$ holds, for all $s\in S$.*

Proof (outline). The proof goes by induction on the complexity of ϕ. Cases 1–4 of Definition 3.5 are trivial. Case 5 follows directly from Lemma 3.3. Observing that $[\![\boxdot\phi]\!]_\rho(s)=0$ if $s\not\rightarrow$ and $[\![\boxdot\phi]\!]_\rho(s)=[\![\Box\phi]\!]_\rho(s)$ otherwise, also Case 6 is a consequence of Lemma 3.3. Consider cases 7 and 8. The encoding is of the form $\mu X.(F\sqcup(G\sqcap H(X))$, where F and G (by induction hypothesis) and $H(X)$ (by Proposition 3.2) are all $\{0,1\}$-valued. Therefore the functor $f\mapsto[\![F\sqcup(G\sqcap H(X))]\!]_{\rho[f/X]}$ maps $\{0,1\}$-valued functions to $\{0,1\}$-valued functions and has only $\{0,1\}$-valued fixed-points. It then follows by Remark 3.4 that the correctness of the encoding for these two cases can be proved with the standard technique used to prove the correctness of the encoding of CTL into Kozen's μ-calculus (see, e.g., [24]). Consider Case 9. It is immediate to verify that $\bigsqcup_\sigma\{m^s_\sigma(U)\}$, where $U=(\!|\circ\phi|\!)_\rho=\bigcup\{U_{\{s.t\}}\mid t\in(\!|\phi|\!)_\rho\}$, is equal (by induction hypothesis) to $[\![\Diamond\mathbf{E}(\phi)]\!]_\rho(s)$. The desired equality $(\!|\mathbb{P}^\exists_{\bowtie q}\circ\phi|\!)_\rho=[\![\mathbb{P}_{\bowtie q}\Diamond\mathbf{E}(\phi)]\!]_\rho$ then follows by Proposition 3.2. Case 10 is similar. The two cases 11 and 12 are similar, thus we just consider case 11. Let $\phi=\mathbb{P}^\exists_{\bowtie q}(\psi)$ and $\psi=\phi_1\mathscr{U}\phi_2$. We denote with Ψ the set of paths $(\!|\psi|\!)_\rho$. Denote by $F(X)$ the formula $\mathbf{E}(\phi_2)\sqcup(\mathbf{E}(\phi_1)\sqcap\Diamond X)$. It is clearly sufficient to prove that the equality $\bigsqcup_\sigma\{m^s_\sigma(\Psi)\}=[\![\mu X.F(X)\big)]\!]_\rho(s)$ holds. Note that $\mu X.F(X)$ can be expressed as an equivalent qLμ formulas by substituting the closed subformulas $\mathbf{E}(\phi_1)$ and $\mathbf{E}(\phi_2)$ with two fresh atomic predicates P_i with interpretations $\rho(P_i)=[\![\mathbf{E}(\phi_i)]\!]$. The equality can then be proved by simple arguments based on the game-semantics of qLμ (see, e.g., [18] and [20]), similar to the ones used to prove that the Kozen's μ-calculus formula $\mu X.(P_2\vee(P_1\wedge\Diamond X))$ has the same denotation of the CTL formula $\exists(P_1\mathscr{U}P_2)$ (see, e.g., [24]). □

4 Łukasiewicz μ-terms

The aim of the second half of the paper is to show how to compute the (rational) denotational value of a Łμ formula at any state in a finite rational probabilistic transition system. In this section, we build the main machinery for doing this, based on a system of fixed-point terms for defining monotone functions from $[0,1]^n$ to $[0,1]$. The syntax of *(Łukasiewicz) μ-terms* is specified by the grammar:

$$t ::= x \mid qt \mid t\sqcup t \mid t\sqcap t \mid t\oplus t \mid t\odot t \mid \mu x.t \mid \nu x.t$$

Again, q ranges over rationals in $[0,1]$. As expected, the μ and ν operators bind their variables. We write $t(x_1,\ldots,x_n)$ to mean that all free variables of t are contained in $\{x_1,\ldots,x_n\}$.

The *value* $t(\vec{r})$ (we eschew semantic brackets) of a μ-term $t(x_1,\ldots,x_n)$ applied to a vector $(r_1,\ldots,r_n)\in[0,1]^n$ is defined inductively in the obvious way, cf. Definition 2.5. (Indeed, μ-terms form a fragment of Łμ of formulas whose value is independent of the transition system in which they are interpreted.)

In Section 6, the model-checking task will be reduced to the problem of computing the value of μ-terms. The fundamental property that allows such values to be computed is that, for any μ-term $t(x_1,\ldots,x_n)$ and vector of rationals $(q_1,\ldots,q_n)$, the value of $t(\vec{q})$ is rational and can be computed from

t and q. One way of establishing this result is by a simple reduction to the first-order theory of rational linear arithmetic, which provides an indirect means of computing the value of $t(\vec{q})$. The current section presents a brief outline of this approach. After this, in Section 5, we provide an alternative direct algorithm for computing $t(\vec{q})$.

A *linear expression* in variables $x_1,\ldots,x_n$ is an expression

$$q_1x_1+\cdots+q_nx_n+q$$

where $q_1,\ldots,q_n,q$ are real numbers. In the sequel, we only consider *rational* linear expressions, in which $q_1,\ldots,q_n,q$ are all rational, and we henceforth assume this property without mention. We write $e(x_1,\ldots,x_n)$ if e is a linear expression in $x_1,\ldots,x_n$, in which case, given real numbers $r_1,\ldots,r_n$, we write $e(\vec{r})$ for the value of the expression when the variables $\vec{x}$ take values $\vec{r}$. We also make use of the closure of linear expressions under substitution: given $e(x_1,\ldots,x_n)$ and $e_1(y_1,\ldots,y_m),\ldots,e_n(y_1,\ldots,y_m)$, we write $e(e_1,\ldots,e_n)$ for the evident substituted expression in variables $y_1,\ldots,y_m$ (which is defined formally by multiplying out and adding coefficients).

The first-order theory of *rational linear arithmetic* has linear expressions as terms, and strict and non-strict inequalities between linear expressions,

$$e_1<e_1 \qquad e_1\leq e_2 \ , \tag{1}$$

as atomic formulas. Equality can be expressed as the conjunction of two non-strict inequalities and the negation of an atomic formula can itself be expressed as an atomic formula. The truth of a first-order formula is given via its interpretation in the reals, or equivalently in the rationals since the inclusion of the latter in the former is an elementary embedding. The theory enjoys quantifier elimination [8].

Proposition 4.1. *For every Łukasiewicz μ-term $t(x_1,\ldots,x_n)$, its graph $\{(\vec{x},y)\in[0,1]^{n+1} \mid t(\vec{x})=y\}$ is definable by a formula $F_t(x_1,\ldots,x_n,y)$ in the first-order theory of rational linear arithmetic, where F_t is computable from t.*

Proof. The proof is a straightforward induction on the structure of t. We consider two cases, in order to illustrate the simple manipulations used in the construction of F_t.

If t is $t_1\oplus t_2$ then F_t is the formula

$$\exists z_1,z_2.\, F_{t_1}(\vec{x},z_1)\wedge F_{t_2}(\vec{x},z_2)\wedge((z_1+z_2\leq 1\wedge z=z_1+z_2)\vee(1\leq z_1+z_2\wedge z=1))$$

If t is $\mu x_{n+1}.t'$ then F_t is the formula

$$F_{t'}(x_1,\ldots,x_n,y,y)\wedge\forall z.\, F_{t'}(x_1,\ldots,x_n,z,z)\rightarrow y\leq z \ .$$

□

Proposition 4.1 provides the following method of computing the value $t(\vec{q})$ of μ-term $t(x_1,\ldots,x_n)$ at a rational vector $(q_1,\ldots,q_n)\in[0,1]^n$. First construct $F_t(x_1,\ldots,x_n,y)$. Next, perform quantifier elimination to obtain an equivalent quantifier-free formula $G_t(x_1,\ldots,x_n,y)$, and consider its instantiation $G_t(q_1,\ldots,q_n,y)$ at $\vec{q}$. (Alternatively, obtain an equivalent formula $G_t^{\vec{q}}(y)$ by performing quantifier elimination on $F_t(q_1,\ldots,q_n,y)$.) By performing obvious simplifications of atomic formulas in one variable, $G_t(q_1,\ldots,q_n,y)$ reduces to a boolean combination of inequalities each having one of the following forms

$$y\leq q \qquad y<q \qquad y\geq q \qquad y>q \ .$$

By the correctness of G_t there must be a unique rational satisfying the boolean combination of constraints, and this can be extracted in a straightforward way from $G_t(q_1,\ldots,q_n,y)$.

We give a crude (but sufficient for our purposes) complexity analysis of the above procedure. In general, for a μ-term t of length u containing v fixed points, the length of F_t is bounded by $2^v uc$, for some constant c. The quantifier-elimination procedure in [8], when given a formula of length l as input produces a formula of length at most 2^{dl} as output, for some constant d, and takes time at most $2^{2^{d'l}}$. Thus the length of the formula $G_t(x_1,\ldots,x_n,y)$ is bounded by $2^{2^v ucd}$, and the computation time for $t(\vec{q})$ is $\mathrm{O}\big(2^{2^{2^v ucd'}}\big)$, using a unit cost model for rational arithmetic.

5 A direct algorithm for evaluating μ-terms

Our direct approach to computing the values of μ-terms is based on a simple explicit representation of the functions defined by such terms. A *conditioned linear expression* is a pair, written $C \vdash e$, where e is a linear expression, and C is a finite set of strict and non-strict inequalities between linear expressions; i.e., each element of C has one of the forms in (1). We write $C(\vec{r})$ for the conjunction of the inequations obtained by instantiating $\vec{r}$ for $\vec{x}$ in C. Clearly, if $\vec{q}$ is a vector of rationals then it is decidable if $C(\vec{q})$ is true or false. The intended meaning of a conditioned linear expression $C \vdash e$ is that it denotes the value $e(\vec{r})$ when applied to a vector of reals $\vec{r}$ for which $C(\vec{r})$ is true, otherwise it is undefined. A basic property we exploit in the sequel is that every conditioning set $C(x_1,\ldots,x_n)$ defines a convex subset $\{(r_1,\ldots,r_n) \mid C(\vec{r})\}$ of $\mathbb{R}^n$.

Let $\mathscr{F}$ be a *system* (i.e., finite set) of conditioned linear expresssions in variables $x_1,\ldots,x_n$. We say that $\mathscr{F}$ *represents* a function $f\colon [0,1]^n \to [0,1]$ if the following conditions hold:

1. For all $d_1,\ldots,d_n \in [0,1]$, there exists a conditioned linear expression $(C \vdash e) \in \mathscr{F}$ such that $C(\vec{d})$ is true, and
2. for all $d_1,\ldots,d_n \in [0,1]$, and every conditioned linear expression $(C \vdash e) \in \mathscr{F}$, if $C(\vec{d})$ is true then $e(\vec{d}) = f(\vec{d})$.

Note that, for two conditioned linear expressions $(C_1 \vdash e_1), (C_2 \vdash e_2) \in \mathscr{F}$, we do not require different conditioning sets C_1 and C_2 to be disjoint. However, e_1 and e_2 must agree on any overlap.

Obviously, the function represented by a system of conditioned linear expressions is unique, when it exists. But not every system represents a function. One could impose syntactic conditions on a system to ensure that it represents a function, but we shall not pursue this.

While conditioned linear expressions provide a syntax more directly tailored to expressing functions than general logical formulas, their expressivity in this regard coincides with rational linear arithmetic.

Proposition 5.1. *A function $f\colon [0,1]^n \to [0,1]$ is representable by a system of conditioned linear expressions if and only if its graph $\{(\vec{x},y) \in [0,1]^{n+1} \mid f(\vec{x}) = y\}$ is definable by a formula $F(x_1,\ldots,x_n,y)$ in the first-order theory of rational linear arithmetic. Moreover, a defining formula and a representing system of conditioned linear equations can each be computed from the other.*

We believe this result to be folklore. The proof is a straightforward application of quantifier elimination.

Combining Propositions 4.1 and 5.1 we obtain:

Corollary 5.2. *For every Łukasiewicz μ-term $t(x_1,\ldots,x_n)$, the function*

$$\vec{r} \mapsto t(\vec{r})\colon [0,1]^n \to [0,1]$$

is representable by a system of conditioned linear expressions in variables $x_1,\ldots,x_n$. Furthermore a representing system can be computed from t.

The computation of a representing system for t via quantifier elimination, provided by the proofs of Propositions 4.1 and 5.1, is indirect. The goal of this section is to present an alternative algorithm for calculating the value $t(\vec{r})$ of a μ-term at rationals $r_1,\ldots,r_n \in [0,1]$, which is directly based on manipulating conditioned linear expressions. Rather than computing an entire system of conditioned linear expressions representing t, the algorithm works locally to provide a single conditioned expression that applies to the input vector $\vec{r}$.

The algorithm takes, as input, a μ-term $t(x_1,\ldots,x_n)$ and a vector of rationals $(r_1,\ldots,r_n) \in [0,1]^n$, and returns a conditioned linear expression $C \vdash e$, in variables $x_1,\ldots,x_n$, with the following two properties.

(P1) $C(\vec{r})$ is true.

(P2) For all $s_1,\ldots,s_n \in \mathbb{R}$, if $C(\vec{s})$ is true then $s_1,\ldots,s_n \in [0,1]$ and $e(\vec{s}) = t(\vec{s})$.

It follows that $e(\vec{r}) = t(\vec{r})$, so e can indeed be used to compute the value $t(\vec{r})$.

5.1 The algorithm

The algorithm takes, as input, a μ-term $t(x_1,\ldots,x_n)$ and a vector of rationals $(r_1,\ldots,r_n) \in [0,1]^n$, and returns a conditioned linear expression $C \vdash e$, in variables $x_1,\ldots,x_n$, with the properties (P1) and (P2) above. For the purposes of the correctness proof in Section 5.3, it is convenient to consider the running of the algorithm in the more general case that $r_1,\ldots,r_n$ are arbitrary real numbers in $[0,1]$. This more general algorithm can be understood as an algorithm in the Real RAM (a.k.a. BSS) model of computation [3]. When the input vector is rational, all real numbers encountered during execution of the algorithm are themselves rational, and so the general Real RAM algorithm specialises to a *bona fide* (Turing Machine) algorithm in this case. Moreover, even in the case of irrational inputs, all linear expressions constructed in the course of the algorithm are rational.

The algorithm works recursively on the structure of the term t. We present illustrative cases for terms $t_1 \oplus t_2$ and $\mu x_{n+1}.t'$. The latter is the critical case. The algorithm for $\nu x_{n+1}.t'$ is an obvious dualization.

If t is $t_1 \oplus t_2$ then recursively compute $C_1 \vdash e_1$ and $C_2 \vdash e_2$. If $e_1(\vec{r}) + e_2(\vec{r}) \leq 1$ then return

$$C_1, C_2, e_1 + e_2 \leq 1 \vdash e_1 + e_2 \ .$$

Otherwise, return

$$C_1, C_2, e_1 + e_2 \geq 1 \vdash 1 \ .$$

In the case that t is $\mu x_{n+1}.t'$, enter the following loop starting with $D = \emptyset$ and $d = 0$.

Loop: At the entry of the loop we have a finite set D of inequalities between linear expressions in $x_1,\ldots,x_n$, and we have a linear expression $d(x_1,\ldots,x_n)$. The loop invariant that applies is:

(I1) $D(\vec{r})$ is true; and

(I2) for all $\vec{s} \in [0,1]^n$, if $D(\vec{s})$ then $d(\vec{s}) \leq (\mu x_{n+1}.t')(\vec{s})$.

We think of D as constraints propagated from earlier iterations of the loop, and of d as the current approximation to the least fixed point subject to the constraints.

Recursively compute $t'(x_1,\ldots,x_{n+1})$ at $(\vec{r}, d(\vec{r}))$ as $C \vdash e$, where e has the form:

$$q_1 x_1 + \cdots + q_n x_n + q_{n+1} x_{n+1} + q \ . \tag{2}$$

In the case that $q_{n+1} \neq 1$, define the linear expression:

$$f := \frac{1}{1-q_{n+1}} \left(q_1 x_1 + \cdots + q_n x_n + q \right) \ . \tag{3}$$

Test if $C(\vec{r}, f(\vec{r}))$ is true. If it is, exit the loop and return:

$$D \cup C(x_1, \ldots, x_n, d(x_1, \ldots, x_n)) \cup C(x_1, \ldots, x_n, f(x_1, \ldots, x_n)) \vdash f \tag{4}$$

as the result of the algorithm for $\mu x.t'$ at $\vec{r}$. Otherwise, if $C(\vec{r}, f(\vec{r}))$ is false, define $N(x_1, \ldots, x_n)$ to be the negation of the inequality $e_1(x_1, \ldots, x_n, f(x_1, \ldots x_n)) \lhd e_2(x_1, \ldots, x_n, f(x_1, \ldots x_n))$ (using $\lhd$ to stand for either $<$ or $\leq$), where $e_1(x_1, \ldots, x_{n+1}) \lhd e_2(x_1, \ldots, x_{n+1})$ is a chosen inequality in C for which $e_1(\vec{r}, f(\vec{r})) \lhd e_2(\vec{r}, f(\vec{r}))$ is false, and go to **find next approximation** below.

In the case that $q_{n+1} = 1$, test the equality $q_1 r_1 + \cdots + q_n r_n + q = 0$. If true, exit the loop with result:

$$D \cup C(x_1, \ldots, x_n, d(x_1, \ldots, x_n)) \cup \{q_1 x_1 + \cdots + q_n x_n + q = 0\} \vdash d \ . \tag{5}$$

If instead $q_1 r_1 + \cdots + q_n r_n + q \neq 0$, choose $N(x_1, \ldots, x_n)$ to be whichever of the inequalities

$$q_1 x_1 + \cdots + q_n x_n + q \ < \ 0 \qquad 0 \ < \ q_1 x_1 + \cdots + q_n x_n + q$$

is true for $\vec{r}$, and proceed with **find next approximation** below.

Find next approximation: Arrange the inequalities in C so they have the following structure.

$$C' \cup \{x_{n+1} > a_i\}_{1 \leq i \leq l'} \cup \{x_{n+1} \geq a_i\}_{l' < i \leq l} \cup \{x_{n+1} \leq b_i\}_{1 \leq i \leq m'} \cup \{x_{n+1} < b_i\}_{m' < i \leq m} \tag{6}$$

such that the only variables in the inequalities C', and linear expressions a_i, b_i are $x_1, \ldots, x_n$. Choose j with $1 \leq j \leq m$ such that $b_j(\vec{r}) \leq b_i(\vec{r})$ for all i with $1 \leq i \leq m$. Then go back to **loop**, taking

$$D \cup C(x_1, \ldots, x_n, d(x_1, \ldots, x_n)) \cup \{N(x_1, \ldots, x_n)\} \cup \{b_j \leq b_i \mid 1 \leq i \leq m\} \qquad e(\vec{x}, b_j(\vec{x})) \tag{7}$$

to replace D and d respectively.

5.2 A simple example

Consider the Łμ term $t = \mu x.(\mathbb{P}_{\geq \frac{1}{2}} x \sqcup \frac{1}{2})$, where $\mathbb{P}_{\geq \frac{1}{2}} x$ is the macro formula as in Definition 3.1, that is $\mathbb{P}_{\geq \frac{1}{2}} x = \mathbb{P}_{=1}(x \oplus \frac{1}{2}) = \nu y.(y \odot (x \oplus \frac{1}{2}))$. Thus,

$$t = \mu x.\Big(\nu y.\big(y \odot (x \oplus \frac{1}{2})\big) \sqcup \frac{1}{2} \Big)$$

Here, $t'(x) = \nu y.\big(y \odot (x \oplus \frac{1}{2})\big) \sqcup \frac{1}{2}$ is a discontinuous function, and the value of t is 1.

We omit giving a detailed simulation of the algorithm on the subexpression $t'(x)$ at $x = r$. The result it produces, however, is $\{0 \leq x < \frac{1}{2}\} \vdash \frac{1}{2}$ if $r < \frac{1}{2}$, and $\{\frac{1}{2} \leq x \leq 1\} \vdash 1$ if $r \geq \frac{1}{2}$.

We run the algorithm on input $\mu x.t'(x)$. Set $D = \emptyset$ and $d = 0$. Calculating $t'(x)$ at $x = 0$ we obtain $C \vdash e$ as $\{0 \leq x < \frac{1}{2}\} \vdash \frac{1}{2}$. We now need to calculate $f := \frac{1}{1-0}(\frac{1}{2}) = \frac{1}{2}$. The constraint $C(\frac{1}{2})$ does not hold. Thus we need to improve the approximation $d = 0$. Since $e = \frac{1}{2}$ is constant, the next approximation is $\frac{1}{2}$. The new set of constraints is still the emptyset. Thus we iterate the algorithm with $D = \emptyset$ and $d = \frac{1}{2}$. Calculating $t'(x)$ at $x = \frac{1}{2}$ produces $C \vdash e$ as $\{\frac{1}{2} \leq x \leq 1\} \vdash 1$. Compute $f := \frac{1}{1-0}(1) = 1$. Since $C(1)$ holds, the algorithm terminates with $\emptyset \vdash 1$, as desired.

5.3 Correctness of the algorithm

Theorem 5.3. *Let $t(x_1,\dots,x_n)$ be any Łukasiewicz μ-term. Then, for every input vector $(r_1,\dots,r_n) \in [0,1]^n$, the above (Real RAM) algorithm terminates with a conditioned linear expression $C_{\vec{r}} \vdash e_{\vec{r}}$ satisfying properties (P1) and (P2). Moreover, the set of all possible resulting conditioned linear expressions*

$$\{C_{\vec{r}} \vdash e_{\vec{r}} \mid \vec{r} \in [0,1]^n\} \tag{8}$$

is finite, and thus provides a representing system for the function $t\colon [0,1]^n \to [0,1]$.

Before the proof it is convenient to introduce some terminology associated with the properties stated in the theorem. For a μ-term t, we call the cardinality of the set (8) of possible results, $C_{\vec{r}} \vdash e_{\vec{r}}$, the *basis size*, and we call the maximum number of inequalities in any $C_{\vec{r}}$ the *condition size*.

Proof. By induction on the structure of t. We verify the critical case when t is $\mu x_{n+1}.t'$.

We show first that the loop invariants (I1), (I2) guarantee that any result returned via (4) or (5) satisfies (P1) and (P2). By induction hypothesis, the recursive computation of $t'(x_1,\dots,x_{n+1})$ at $(\vec{r},d(\vec{r}))$ as $C \vdash e$, where e has the form $q_1x_1+\cdots+q_nx_n+q_{n+1}x_{n+1}+q$ as in (2), satisfies: $C(\vec{r},d(\vec{r}))$; and, for all $s_1,\dots,s_{n+1} \in \mathbb{R}$, if $C(s_1,\dots,s_{n+1})$ then $\vec{s} \in [0,1]^n$ and $t'(s_1,\dots,s_{n+1}) = e(s_1,\dots,s_{n+1})$.

In the case that $q_{n+1} \neq 1$, the linear expression f, defined in (3), maps any $s_1,\dots,s_n \in \mathbb{R}$ to the unique solution $f(\vec{s})$ to the equation $x_{n+1} = e(s_1,\dots,s_n,x_{n+1})$ in $\mathbb{R}$. Suppose that $D(\vec{s})$ holds. Then, by loop invariant (I2), $d(\vec{s}) \leq (\mu x_{n+1}.t')(\vec{s})$. Suppose also that $C(\vec{s},f(\vec{s}))$. Then $t'(\vec{s},f(\vec{s})) = e(\vec{s},f(\vec{s})) = f(\vec{s})$, i.e., $f(\vec{s})$ is a fixed point of $x_{n+1} \mapsto t'(\vec{s},x_{n+1})$; whence, $(\mu x_{n+1}.t')(\vec{s}) \leq f(\vec{s})$. Suppose, finally, that $C(\vec{s},d(\vec{s}))$ also holds. Then, because both $C(\vec{s},d(\vec{s}))$ and $C(\vec{s},f(\vec{s}))$, and $d(\vec{s}) \leq (\mu x_{n+1}.t')(\vec{s}) \leq f(\vec{s})$, we have, by the convexity of constraints, that $t'(\vec{s},s_{n+1}) = e(\vec{s},s_{n+1})$ for all $s_{n+1} \in [d(\vec{s}),f(\vec{s})]$. So $f(\vec{s})$ is the unique fixed-point of $x_{n+1} \mapsto t'(\vec{s},x_{n+1})$ on $[d(\vec{s}),f(\vec{s})]$. Since, $d(\vec{s}) \leq (\mu x_{n+1}.t')(\vec{s})$, we have $f(\vec{s}) = (\mu x_{n+1}.t')(\vec{s})$. This argument justifies that the conditioned linear expression of (4) satisfies (P2). It satisfies (P1) just if $C(\vec{r},f(\vec{r}))$, which is exactly the condition under which (4) is returned as the result.

In the case that $q_{n+1} = 1$ then, for any $s_1,\dots,s_n \in \mathbb{R}$, the equation $x_{n+1} = e(s_1,\dots,s_n,x_{n+1})$ has a solution if and only if $q_1s_1+\cdots+q_ns_n+q = 0$, in which case any $x_{n+1} \in \mathbb{R}$ is a solution. Suppose that $q_1s_1+\cdots+q_ns_n+q = 0$ and $C(\vec{s},d(\vec{s}))$ both hold. Then $t'(s_1,\dots,s_n,d(\vec{s})) = e(\vec{s},d(\vec{s})) = d(\vec{s})$, so $d(\vec{x})$ is a fixed point of $x_{n+1} \mapsto t'(\vec{s},x_{n+1})$. If also $D(\vec{s})$ holds then, by loop invariant (I2), $d(\vec{x}) = (\mu x_{n+1}.t')(\vec{s})$. We have justified that the conditioned linear expression of (5) satisfies (P2). It satisfies (P1) just if $q_1r_1+\cdots+q_nr_n+q = 0$, which is exactly the condition under which (5) is returned as the result.

Next we show that the loop invariants are preserved through the computation. Properties (I1) and (I2) are trivially satisfied by the initial values $D = \emptyset$ and $d = 0$. We must show that they are preserved when D and d are modified via (7), which happens when execution passes to **find next approximation**. In this subroutine, the inequalities in C are first arranged as in (6) where, as $C(\vec{r},d(\vec{r}))$, we must have $m \geq 1$, as otherwise $C(\vec{r},s)$ would hold for all real $s \geq d(\vec{r})$, contradicting that $C(\vec{r},s)$ implies $s \in [0,1]$. (Similarly, $l \geq 1$.) Thus there indeed exists j with $1 \leq j \leq m$ such that $b_j(\vec{r}) \leq b_i(\vec{r})$ for all i with $1 \leq i \leq m$. It is immediate that the constraints in the modified D of (7) are true for $\vec{r}$. Thus (I1) is preserved. To show (I2), suppose $s_1,\dots,s_n$ satisfy the constraints, i.e.,

$$D(\vec{s}) \qquad C(\vec{s},d(\vec{s})) \qquad N(\vec{s}) \qquad \{b_j(\vec{s}) \leq b_i(\vec{s}) \mid 1 \leq i \leq m\}\ .$$

Defining $r' = (\mu x_{n+1}.t')(\vec{s})$, by (I2) for D,d we have $d(\vec{s}) \leq r'$. We must show that $e(\vec{s},b_j(\vec{s})) \leq r'$. By the definition of $N(x_1,\dots,x_n)$, in either the $q_{n+1} \neq 1$ or $q_{n+1} = 1$ case, $N(\vec{s})$ implies that $C(\vec{s},r')$ does not hold. Because $C(\vec{s},d(\vec{s}))$ and by the choice of j, it holds that $C(\vec{s},s)$, for all $s \in [0,1]$ such that $s = d(\vec{s})$ or

$d(\vec{s}) < s < b_j(\vec{s})$. Since $C(\vec{s},r')$ is false and $d(\vec{s}) \leq r'$, it follows from the convexity of the conditioning set C that, for every s with $s = d(\vec{s})$ or $d(\vec{s}) < s < b_j(\vec{s})$, we have $s < r'$. Whence, since r' is the least prefixed point for $x_{n+1} \mapsto t'(\vec{s},x_{n+1})$, also $s < t'(\vec{s},s) \leq r'$, i.e.,

$$s < e(\vec{s},s) \leq r' \quad . \tag{9}$$

Thus, $e(\vec{s},b_j(\vec{s})) = \sup\{e(\vec{s},s) \mid s = d(\vec{s}) \text{ or } d(\vec{s}) \leq s < b_j(\vec{s})\} \leq r'$. Thus, $e(\vec{s},b_j(\vec{s})) \leq r'$, i.e., it is an approximation to the fixed point. Moreover, it is a good new approximation to choose in the sense that:

$$d(\vec{s}) < e(\vec{s},b_j(\vec{s})) \text{ and } \text{not } C(\vec{s},e(\vec{s},b_j(\vec{s}))) \quad . \tag{10}$$

The former holds because $d(\vec{s}) < e(\vec{s},d(\vec{s}))$, by (9), and $d(\vec{s}) \leq b_j(\vec{s})$. The latter because if $C(\vec{s},e(\vec{s},b_j(\vec{s})))$ then, in particular, $e(\vec{s},b_j(\vec{s})) \leq b_j(\vec{s})$, so $b_j(\vec{s}) = e(\vec{s},b_j(\vec{s})) = r'$, contradicting that not $C(\vec{s},r')$.

To show termination, by induction hypothesis, collecting all possible results of running the algorithm on t' produces a representing system for $t'\colon [0,1]^{n+1} \to [0,1]$:

$$C_1 \vdash e_1 \quad \ldots \quad C_{k'} \vdash e_{k'} \quad , \tag{11}$$

where k' is the basis size of t'. We now analyse the execution of the algorithm for $\mu x_{n+1}.t'$ on a given input vector $(r_1,\ldots,r_n)$. On iteration number i, the loop is entered with constraints D_i and approximation d_i (where $D_1 = \emptyset$ and $d_1 = 0$), after which the recursive call to the algorithm for t' yields one of the conditioned linear expressions, $C_{k_i} \vdash e_{k_i}$, from (11) above, such that $C_{k_i}(\vec{r},d_i(\vec{r}))$ holds. Then, depending on conditions involving only $C_{k_i} \vdash e_{k_i}$ and $\vec{r}$, either a result is returned, or D_{i+1} and d_{i+1} are constructed for the loop to be repeated. By (10), at iteration $i+1$ of the loop, we have $d_{i+1}(\vec{r}) > d_i(\vec{r})$ and also $C_{k_i}(\vec{r},d_{i+1}(\vec{r}))$ is false. Since each conditioning set is convex, it follows that no C_j can occur twice in the list $C_{k_1},C_{k_2},\ldots$. Hence the algorithm must exit the loop after at most k' iterations. Therefore, the computation for $\mu x.t'$ at $\vec{r}$ terminates.

It remains to show that the algorithm for $\mu x.t'$ produces only finitely many conditioned linear expressions $C_{\vec{r}} \vdash e_{\vec{r}}$. The crucial observation is that the vector $\vec{r}$ is used only to determine the control flow of the algorithm, i.e., which branches of conditional statements are followed, the choices made in selecting N and b_j in (7), and the order in which the different $C_j \vdash e_j$, from (11) are visited (given by the sequence $k_1,k_2,\ldots$ of values taken by j). Using this, if l' is the condition size of t', then a loose upper bound is that the number of possible results $C_{\vec{r}} \vdash e_{\vec{r}}$ for the algorithm for $\mu x_{n+1}.t'$ is at most $(k'(l')^2)^{k'}$, and the number of inequalities in $C_{\vec{r}}$ is at most $2k'l'$. □

The above proof gives a truly abysmal complexity bound for the algorithm. Let the basis and condition size for the term $t'(x_1,\ldots,x_{n+1})$ be k' and l' respectively. Then, as in the proof, the basis and condition size for $\mu x_{n+1}.t'$ are respectively bounded by:

$$k \leq (k'(l')^2)^{k'} \text{ and } l \leq 2k'l' \quad .$$

Using these bounds, the basis and condition size have non-elementary growth in the number of fixed points in a term t.

5.4 Comparison

According to the crude complexity analyses we have given, the evaluation of Łukasiewicz μ-terms via rational linear arithmetic is (in having doubly- and triply-exponential space and time complexity bounds)

preferable to the (non-elementary space and hence time) evaluation via the direct algorithm. Nevertheless, we expect the direct algorithm to work better than this in practice. Indeed, a main motivating factor in the design of the direct algorithm is that the algorithm for $\mu x_{n+1}.t'$ only explores as much of the basis set for t' as it needs to, and does so in an order that is tightly constrained by the monotone improvements made to the approximating d expressions along the way. In contrast, the crude complexity analysis is based on a worst-case scenario in which the algorithm is assumed to visit the entire basis for t', and, moreover, to do so, for different input vectors $\vec{r}$, in every possible order for visiting the different basis sets. Perhaps better bounds can be obtained by a more careful analysis of the algorithm.

6 Model checking

Let ϕ be a closed Łμ formula and $(S,\to)$ a finite rational PNTS. We wish to compute the value $[\![\phi]\!](s)$ at any given state $s \in S$. We do this by effectively producing a closed μ-term $t_s(\phi)$, with the property that $t_s(\phi) = [\![\phi]\!](s)$, whence the rational value of $[\![\phi]\!](s)$ can be calculated by the algorithm in Section 5.

We assume, without loss of generality, that all fixed-point operators in ϕ bind distinct variables. Let $X_1,\ldots,X_m$ be the variables appearing in ϕ. We write $\sigma_i X_i.\,\psi_i$ for the unique subformula of ϕ in which X_i is bound. The strict (i.e., irreflexive) *domination* relation $X_i \rhd X_j$ between variables is defined to mean that $\sigma_j X_j.\,\psi_j$ occurs as a subformula in ψ_i.

Suppose $|S| = n$. For each $s \in S$, we translate ϕ to a μ-term $t_s(\phi)$ containing at most mn variables $x_{i,s'}$, where $1 \le i \le m$ and $s' \in S$. The translation is defined using a more general function t_s^Γ, defined on subformulas of ϕ, where $\Gamma \subseteq \{1,\ldots,m\} \times S$ is an auxiliary component keeping track of the states at which variables have previously been encountered. Given Γ and $(i,s) \in \{1,\ldots,m\} \times S$, we define:

$$\Gamma \rhd (i,s) \;=\; (\Gamma \cup \{(i,s)\}) \backslash \{(j,s') \in \Gamma \mid X_i \rhd X_j\} \;\;.$$

This operation is used in the definition below to 'reset' subordinate fixed-point variables whenever a new variable that dominates them is declared.

$$\begin{aligned}
t_s^\Gamma(X_i) &= \begin{cases} x_{i,s} & \text{if } (i,s) \in \Gamma \\ \sigma_i x_{i,s}.\, t_s^{\Gamma \rhd (i,s)}(\psi_i) & \text{otherwise} \end{cases} \\
t_s^\Gamma(P) &= \underline{\rho(P)(s)} \\
t_s^\Gamma(\overline{P}) &= \underline{1-\rho(P)(s)} \\
t_s^\Gamma(q\,\phi) &= q\, t_s^\Gamma(\phi) \\
t_s^\Gamma(\phi_1 \bullet \phi_2) &= t_s^\Gamma(\phi_1) \bullet t_s^\Gamma(\phi_2) \qquad \bullet \in \{\sqcup, \sqcap, \oplus, \odot\} \\
t_s^\Gamma(\Diamond\phi) &= \bigsqcup_{s \to d} \bigoplus_{s' \in S} d(s')\, t_{s'}^\Gamma(\phi) \\
t_s^\Gamma(\Box\phi) &= \bigsqcap_{s \to d} \bigoplus_{s' \in S} d(s')\, t_{s'}^\Gamma(\phi) \\
t_s^\Gamma(\sigma_i X_i.\,\psi_i) &= \sigma_i x_{i,s}.\, t_s^{\Gamma \cup \{(i,s)\}}(\psi_i)
\end{aligned}$$

This is well defined because changing from Γ to $\Gamma \rhd (i,s)$ or to $\Gamma \cup \{(i,s)\}$ strictly increases the function

$$i \mapsto |\{(i,s) \mid (i,s) \in \Gamma\}| \colon \{1,\ldots,m\} \to \{0,\ldots,n\}$$

under the lexicographic order on functions relative to $\rhd$.

Proposition 6.1. *For any closed Łμ formula ϕ, finite PNTS $(S,\to)$ and $s\in S$, it holds that $[\![\phi]\!](s) = t_s^{\emptyset}(\phi)$.*

We omit the laborious proof. It is reminiscent of the reduction of modal μ-calculus model checking to a system of nested boolean fixed-point equations in Section 4 of [17].

7 Related and future work

The first encodings of probabilistic temporal logics in a probabilistic version of the modal μ-calculus were given in [4], where a version **PCTL***, tailored to processes exhibiting probabilistic but not nondeterministic choice, was translated into a non-quantitative probabilisitic variant of the μ-calculus, which included explicit (probabilistic) path quantifiers but disallowed fixed-point alternation.

In their original paper on quantitative μ-calculi [12], Huth and Kwiatkowska attempted a model checking algorithm for alternation-free formulas in the version of Łμ with $\oplus$ and $\odot$ but without $\sqcap$, $\sqcup$ and scalar multiplication. Subsequently, several authors have addressed the problem of computing (sometimes approximating) fixed points for monotone functions combining linear (sometimes polynomial) expressions with min and max operations; see [10] for a summary. However, such work has focused on (efficiently) finding outermost (simultaneous) fixed-points for systems of equations whose underlying monotone functions are continuous. The nested fixed points considered in the present paper give rise to the complication of non-continuous functions, as the example of Section 5.2 demonstrates.

As future work, it is planned to run an experimental comparison of the direct algorithm against the reduction to linear arithmetic. As suggested in Section 5.4, we expect the direct algorithm to work better in practice than the non-elementary upper bound on its complexity, given by our crude analysis, suggests. Furthermore, as a natural generalization of the approximation approach to computing fixed points, the direct algorithm should be amenable to optimizations such as the simultaneous solution of adjacent fixed points of the same kind, and the reuse of previous approximations when applicable due to monotonicity considerations. Unlike the black-box reduction to linear arithmetic, based on quantifier elimination, the linear-constraint-based approach of the direct algorithm should also offer a flexible machinery helpful in the design of optimized procedures for calculating values of particular subclasses of Łμ-terms. An important example is given by the fragment of Łμ capable of encoding **PCTL** (see Remark 3.6).

Our results on Łμ are a contribution towards the development of a robust theory of fixed-point probabilistic logics. The simplicity of the proposed encoding of **PCTL** (see Remark 3.6 above) suggests that the direction we are following is promising. In a follow-up paper, by the first author, it will be shown that the process equivalence characterised by Łukasiewicz μ-calculus is the standard notion of *probabilistic bisimilarity* [23]. Thus the quantitative approach to probabilistic μ-calculi may be considered equally suitable as a mechanism for characterising process equivalence as the non-quantitative μ-calculi advocated for this purpose in [4] and [7].

Further research will have to explore the relations between quantitative μ-calculi such as Łμ and other established frameworks for verification and design of probabilistic systems. Important examples include the *abstract probabilistic automata* of [6], the compositional *assume-guarantee* techniques of [16, 9] and the recent *p-automata* of [13]. In particular, with respect to the latter formalism, we note that the acceptance condition of p-automata is specified in terms of stochastic games whose configurations may have preseeded threshold values whose action closely resembles that of the threshold modalities considered in this work (Definition 3.1). Exploring the relations between p-automata games and Łμ-games [19] could shed light on some underlying fundamental ideas.

Acknowledgements

We thank Kousha Etessami, Grant Passmore, Colin Stirling and the anonymous reviewers for helpful comments and for pointers to the literature.

The first author carried out this work during the tenure of an ERCIM "Alain Bensoussan" Fellowship, supported by the Marie Curie Co-funding of Regional, National and International Programmes (COFUND) of the European Commission.

References

[1] Christel Baier & Joost Pieter Katoen (2008): *Principles of Model Checking*. The MIT Press.

[2] Andrea Bianco & Luca de Alfaro (1995): *Model Checking of Probabilistic and Nondeterministic Systems*. In: *Foundations of Software Technology and Theoretical Computer Science, Lecture Notes in Computer Science* 1026, Springer-Verlag, pp. 499–513, doi:10.1007/3-540-60692-0_70.

[3] Lenore Blum, Mike Shub & Steve Smale (1989): *On a Theory of Computation and Complexity over the Real Numbers: NP-completeness, Recursive Functions and Universal Machines*. *Bulletin of the AMS* 21(1), doi:10.1109/SFCS.1988.21955.

[4] Rance Cleaveland, S. Purushothaman Iyer & Muralidhar Narasimha (1999): *Probabilistic Temporal Logics via the Modal mu-Calculus*. In: *Foundations of Software Science and Computation Structures*, doi:10.1007/3-540-49019-1_20.

[5] Luca de Alfaro & Rupak Majumdar (2004): *Quantitative Solution of omega-Regular Games*. *Journal of Computer and System Sciences, Volume 68, Issue 2*, pp. 374 – 397, doi:10.1016/j.jcss.2003.07.009.

[6] Benoit Delahaye, Joost Pieter Katoen, Kim Larsen, Axel Legay, Mikkel Pedersen, Falak Sher & Andrzej Wasowski (2011): *Abstract Probabilistic Automata*. In: *Proc. of 12th VMCAI*, doi:10.1007/978-3-642-18275-4_23.

[7] Yuxin Deng & Rob van Glabbeek (2010): *Characterising Probabilistic Processes Logically*. In: *Logic for programming, artificial intelligence and reasoning, Lecture Notes in Computer Science* 6397, doi:10.1007/978-3-642-16242-8_20.

[8] Jeanne Ferrante & Charles Rackoff (1975): *A Decision Procedure for the First Order Theory of Real Addition with Order*. *SIAM Journal of Computing* 4(1), pp. 69–76, doi:10.1137/0204006.

[9] Vojtěch Forejt, Marta Kwiatkowska, Gethin Norman, David Parker & Hongyang Qu (2011): *Quantitative multi-Objective Verification for Probabilistic Systems*. In: *Proc. of 14th TACAS*, doi:10.1007/978-3-642-19835-9_11.

[10] Thomas Martin Galwitza & Helmut Seidl (2011): *Solving Systems of Rational Equations through Strategy Iteration*. *ACM Trabnsactions on Programming Languages and Systems* 33(3), doi:10.1145/1961204.1961207.

[11] Petr Hájek (2001): *Metamathematics of Fuzzy Logic*. Springer.

[12] Michael Huth & Marta Kwiatkowska (1997): *Quantitative Analysis and Model Checking*. In: *Proceeding of the 12th Annual IEEE Symposium on Logic in Computer Science*.

[13] Michael Huth, Nir Piterman & Daniel Wagner (2012): *p-Automata: New Foundations for discrete-time Probabilistic Verification*. *Perform. Eval.* 69(7-8), doi:10.1016/j.peva.2012.05.005.

[14] David Janin & Igor Walukiewicz (1996): *On the Expressive Completeness of the Propositional mu-Calculus with Respect to Monadic Second Order Logic*. *Lecture Notes in Computer Science* 1119, pp. 263–277, doi:10.1007/3-540-61604-7_60.

[15] D. Kozen (1983): *Results on the Propositional mu-Calculus*. In: *Theoretical Computer Science*, pp. 333–354, doi:10.1016/0304-3975(82)90125-6.

[16] Marta Kwiatkowska, Gethin Norman, David Parker & Hongyang Qu (2010): *Assume-Guarantee Verification for Probabilistic Systems*. In: *Proceedings of 16th TACAS*, doi:10.1007/978-3-642-12002-2_3.

[17] Angelika Mader (1995): *Modal μ-Calculus, Model Checking and Gauß Elimination*. In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS), LNCS* 1019, pp. 72–88, doi:10.1007/3-540-60630-0_4.

[18] Annabelle McIver & Carroll Morgan (2007): *Results on the Quantitative μ-Calculus qMμ*. *ACM Transactions on Computational Logic* 8(1), doi:10.1145/1182613.1182616.

[19] Matteo Mio (2012): *Game Semantics for Probabilistic μ-Calculi*. Ph.D. thesis, School of Informatics, University of Edinburgh. Permanent URL: http://hdl.handle.net/1842/6223.

[20] Matteo Mio (2012): *On The Equivalence of Denotational and Game Semantics for the Probabilistic μ-Calculus*. *Logical Methods in Computer Science* 8(2), doi:10.2168/LMCS-8(2:7)2012.

[21] Matteo Mio (2012): *Probabilistic Modal μ-Calculus with Independent Product*. *Logical Methods in Computer Science* 8(4), doi:10.2168/LMCS-8(4:18)2012.

[22] Carroll Morgan & Annabelle McIver (1997): *A Probabilistic Temporal Calculus Based on Expectations*. In: *In Lindsay Groves and Steve Reeves, editors, Proc. Formal Methods*, Springer Verlag.

[23] Roberto Segala (1995): *Modeling and Verification of Randomized Distributed Real-Time Systems*. Ph.D. thesis, Laboratory for Computer Science, M.I.T.

[24] Colin Stirling (2001): *Modal and Temporal Logics for Processes*. Springer, doi:10.1007/3-540-60915-6_5.

A Appendix: some omitted proof details

We add detail to the outlined proof of Theorem 3.7, by supplying the omited argument for the equality

$$\bigsqcup_{\sigma}\{m^s_\sigma(\Psi)\} = [\![\mu X.F(X)\big)]\!]_\rho(s) \ \ ,$$

which appears as case 11. Although game semantics provides the most intuitive justification, we instead give a direct denotational proof, in order to avoid introducing game-theoretic machinery.

Expanded proof of Theorem 3.7. Case 11 ($\leq$). We first show that

$$\bigsqcup_{\sigma}\{m^s_\sigma(\Psi)\} \leq [\![\mu X.F(X)\big)]\!]_\rho(s) \tag{12}$$

Define $\Psi_k = \{s_0.s_1.s_2\ldots \mid s_0 = s \text{ and } \exists n \leq k.\big(s_n \in (\!|\phi_2|\!)_\rho \wedge \forall m < n.(s_m \in (\!|\phi_1|\!)_\rho)\big)\}$. Clearly $\Psi = \bigcup_k \Psi_k$. Suppose Inequality 12 does not hold. Then there exists some k and scheduler σ such that

$$m^s_\sigma(\Psi_k) > [\![\mu X.F(X)]\!]_\rho(s) \tag{13}$$

We prove that this is not possible by induction on k. In the $k=0$ case, since we are assuming $m^s_\sigma(\Psi_0) > 0$, it holds that $s \in (\!|\phi_2|\!)_\rho$. By inductive hypothesis on ϕ_2, we know that $[\![\mathbf{E}(\phi_2)]\!](s) = 1$ and this implies that $\mu X.F(X) = 1$, which is a contradiction with the assumed strict inequality 13. Consider the case $k+1$. Note that if $s \in (\!|\phi_2|\!)_\rho$ then, $[\![\mu X.F(X)]\!]_\rho(s) = 1$ as before, contradicting Inequality 13. So assume $s \notin (\!|\phi_2|\!)_\rho$. Since we are assuming $m^s_\sigma(\Psi_{k+1}) > 0$ it must be the case that $s \in (\!|\phi_1|\!)_\rho$. Similarly, $m^s_\sigma(\Psi_{k+1}) > 0$ and $s \notin (\!|\phi_2|\!)_\rho$ imply that $s \not\to$ does not hold. This means (see Definition 2.8) that $\sigma(\{s\})$ is defined. Let $d = \sigma(\{s\})$ and observe that $m^s_\sigma(\Psi_{k+1}) = \sum_{t\in S} d(t) m^t_{\sigma'}(\Psi_k)$, where $\sigma'(s_0,s_1,\ldots,s_n) = \sigma(s,s_0,s_1,\ldots,s_n)$.

By induction on k we know that the inequality $m^t_{\sigma'}(\Psi_k) \leq [\![\mu X.F(X)]\!]_\rho(t)$ holds for every $t \in S$. Thus, by definition of the semantics of $\Diamond$, we obtain $m^s_\sigma(\Psi_k) \leq [\![\Diamond(\mu X.F(X))]\!]_\rho$. Recall that we previously assumed $s \notin (\!|\phi_2|\!)_\rho$ and $s \in (\!|\phi_1|\!)_\rho$. Hence the equality

$$[\![\Diamond(\mu X.F(X))]\!]_\rho(s) = [\![\mathbf{E}(\phi_2) \sqcup (\mathbf{E}(\phi_1) \sqcap (\Diamond \mu X.F(X)))]\!]_\rho(s)$$

holds. The formula on the right is just the unfolding $F(\mu X.F(X))$ of $\mu X.F(X)$. This implies the desired contradiction.

Case 11($\geq$). We now prove that also the inequality

$$\bigsqcup_\sigma \{m^s_\sigma(\psi)\} \geq [\![\mu X.F(X)]\!]_\rho(s) \tag{14}$$

holds. By Knaster-Tarski theorem, $[\![\mu X.F(X)]\!]_\rho = \bigsqcup_\alpha [\![F(X)]\!]^\alpha_\rho$, where α ranges over the ordinals and $[\![F(X)]\!]^\alpha_{\rho^\alpha}$ with $\rho^\alpha = \rho[\bigsqcup_{\beta<\alpha}[\![F(X)]\!]_{\rho^\beta}/X]$. We prove Inequality 14 by showing, by transfinite induction, that for every ordinal α and $\varepsilon > 0$, the inequality

$$\bigsqcup_\sigma \{m^s_\sigma(\psi)\} > [\![\mu X.F(X)]\!]_{\rho^\alpha}(s) - \varepsilon \tag{15}$$

holds, for all $s \in S$. The case for $\alpha = 0$ is immediate since $[\![F]\!]_{\rho^0}(s) > 0$ if and only if $[\![\mathbf{E}(\phi_2)]\!]_\rho(s) = 1$ and this implies $\bigsqcup_\sigma \{m^s_\sigma(\psi)\} = 1$. Consider $\alpha = \beta + 1$. If $[\![\mathbf{E}(\phi_2)]\!]_\rho(s) = 1$ then Inequality 14 holds as above. Thus assume $[\![\phi_2]\!]_\rho(s) = 0$. Note that $[\![F]\!]_{\rho^0}(s) > 0$ only if $s \in [\![\mathbf{E}(\phi_1)]\!]$. Thus assume $[\![\mathbf{E}(\phi_1)]\!]^\beta_\rho(s) = 1$. Under these assumption, $[\![F(X)]\!]_{\rho^\alpha} = [\![\Diamond F(X)]\!]_{\rho^\beta}$ as it is immediate to verify. By definition of the semantics of $\Diamond$ we have:

$$[\![\Diamond F(X)]\!]_{\rho^\beta}(s) = \bigsqcup_{s \to d} \big(\sum_{t \in S} d(t)[\![F(X)]\!]_{\rho^\beta}(t)\big)$$

By induction hypothesis on β we know that for every ε,

$$[\![\Diamond F(X)]\!]_{\rho^\beta}(s) < \bigsqcup_{s \to d} \big(\sum_{t \in S} d(t)\Big(\bigsqcup_\sigma \{m^t_\sigma(\psi)\} + \varepsilon\Big)\big)$$

For each $s \to d$ and σ define σ^d as $\sigma^d(\{s\}) = d$ and $\sigma^d(s.t_0\ldots) = \sigma(t_0\ldots)$. A simple argument shows that

$$\bigsqcup_{s \to d} \big(\sum_{t \in S} d(t)\Big(\bigsqcup_\sigma \{m^t_\sigma(\psi)\} + \varepsilon\Big)\big) = \bigsqcup_{\sigma^d} \{m^s_{\sigma^d}(\psi)\} + \varepsilon$$

and this conclude the proof for the case $\alpha = \beta + 1$. Lastly, the case for α a limit ordinal follows straightforwardly from the inductive hypothesis on $\beta < \alpha$. □

Proof of Proposition 5.1. Suppose we have a system of k conditioned linear expressions representing f. Each conditioned expression $C \vdash e$ is captured by the implication $(\bigwedge C) \to y = e$, so the whole system translates into a conjunction of k such implications. To this conjunction, one need only add the range constraints $0 \leq z$ and $z \leq 1$ for each variable z, as further conjuncts. In this way, the graph is easily expressed as a quantifier free formula. (Since the implications are equivalent to disjunctions of atomic formulas, the resulting formula is naturally in conjunctive normal form.)

Conversely, suppose $F(x_1,\ldots,x_n,y)$ defines the graph of f. By quantifier elimination, we can assume that F is quantifier free and in disjunctive normal form. Then F is a disjunction of conjunctions, where each conjunction, K, can be easily rewritten in the form

$$\Big(\bigwedge C\Big) \wedge \left(\bigwedge_{1 \leq i \leq h} y > a_i\right) \wedge \left(\bigwedge_{1 \leq i \leq k} y \geq b_i\right) \wedge \left(\bigwedge_{1 \leq i \leq l} y \leq c_i\right) \wedge \left(\bigwedge_{1 \leq i \leq m} y < d_i\right) \quad , \tag{16}$$

such that the only variables in the finite set of atomic formulas C, and linear expressions a_i, b_i, c_i, d_i are $x_1, \ldots, x_n$. Since F is the graph of a function, for all reals $r_1, \ldots, r_n$, there is at most one s such that $K(\vec{r}, s)$ holds, and, if it does, then all of $r_1, \ldots, r_n, s$ are in $[0,1]$. Given such an s, we therefore have:

$$\max\{a_i(\vec{r}) \mid 1 \leq i \leq h\} < \max\{b_i(\vec{r}) \mid 1 \leq i \leq k\} = s = \min\{c_i(\vec{r}) \mid 1 \leq i \leq l\} < \min\{d_i(\vec{r}) \mid 1 \leq i \leq m\} \ .$$

A system of conditioned linear expressions for f is thus obtained as follows. For each conjunct K in F, written in the form of (16) above, and each j with $1 \leq j \leq k$, include the conditioned linear expression:

$$C, \{b_j > a_i\}_{1 \leq i \leq h}, \{b_j \geq b_i\}_{1 \leq i \leq k}, \{b_j \leq c_i\}_{1 \leq i \leq l}, \{b_j < d_i\}_{1 \leq i \leq m}, \vdash b_j \ .$$

□

We supplement the proof of Theorem 5.3 with more detail on the bounds on basis and condition size.

Expanded proof of Theorem 5.3. We analyse the control flow in the algorithm for $\mu x_{n+1}.t'$ on a given input vector $(r_1, \ldots, r_n)$. On iteration number i, the loop is entered with constraints D_i and approximation d_i, after which the recursive call to the algorithm for t' yields one of the conditioned linear expressions, $C_{k_i} \vdash e_{k_i}$. Suppose that C_{k_i} and D_i contain u and v inequalities respectively. If the loop is exited producing (4) as result then the resulting $C_{\vec{r}}$ has $2u+v$ inequalities. If it is exited producing (5) as result then $C_{\vec{r}}$ has $u+v+2$ inequalities (where $u+v+2 \leq 2u+v$ because C_{k_i} has to enforce the range constraint $0 \leq x_{n+1} \leq 1$). Otherwise, the algorithm repeats the loop, entering iteration $i+1$ with D_{i+1}, given by (7), having at most $2u+v$ inequalities (N contributes 1 inequality, and there are at most $u-1$ inequalities $b_j \leq b_i$ in (7) since $l \geq 1$).

Therefore, if l' is now maximum number of inequalities occurring in any C_j from (11) (i.e., if it is the condition size for t') the algorithm for $\mu x_{n+1}.t'$ at $\vec{r}$, which runs for at most k' iterations, results in $C_{\vec{r}}$ containing at most $2k'l'$ inequalities.

To bound the number of results $C_{\vec{r}} \vdash e_{\vec{r}}$, we count the possible control flows of the algorithm. At iteration i, the algorithm uses $C_{k_i} \vdash e_{k_i}$ from (11), using which it might terminate with either (4) or (5), or it might repeat the loop, entering iteration $i+1$ with D_{i+1}, given by (7), which can arise from C_{k_i} in a number of ways determined by the possible pairs of choices for N and b_j in (7). In the case that the variable vector $(x_1, \ldots, x_n)$ is empty (i.e., the term $\mu x_{n+1}.t'$ is closed) the constraints in D are redundant (they are simply true inequalities between rathionals) and so can be discarded. In the case that $n \geq 1$, there are at least 2 inequalities in C giving range constraints on x_1, so there are at most l' choices for N ($l'-2$ choices in the case that $q_{n+1} \neq 1$, and 2 in the case $q_{n+1} = 1$). Irrespective of n, there are at most $l'-1$ choices for b_j (taking n into account this can be improved to $l'-2n-1$). Therefore, the execution of the algorithm, is determined by the sequence:

$$k_1, u_1, k_2, u_2, \ldots, k_m, v$$

where: $m \leq k'$ is the number of loop iterations performed; each u_i, where $1 \leq u_i \leq l'(l'-1)$, represents the choice of N and b_j used in the construction of D_{i+1} (7), and v is 1 or 2 according to whether the resulting $C_{\vec{r}} \vdash e_{\vec{r}}$ is returned via (4) or (5). Since each number k_i is distinct, the number of different such sequences is bounded by:

$$2 \sum_{m=1}^{k'} \frac{k'!}{(k'-m)!} (l'(l'-1))^{m-1} \ \leq \ (k'(l')^2)^{k'} \ , \tag{17}$$

where the right-hand-side gives a somewhat loose upper bound. Therefore, the number of possible results $C_{\vec{r}} \vdash e_{\vec{r}}$ for the algorithm for $\mu x_{n+1}.t'$ is at most $(k'(l')^2)^{k'}$. □